Short Note on Complexity of Multi-Value Byzantine Agreement *

Guanfeng Liang and Nitin Vaidya
Department of Electrical and Computer Engineering, and
Coordinated Science Laboratory
University of Illinois at Urbana-Champaign
gliang2@illinois.edu, nhv@illinois.edu

Technical Report

July 27, 2010

*This research is supported in part by Army Research Office grant W-911-NF-0710287. Any opinions, findings, and conclusions or recommendations expressed here are those of the authors and do not necessarily reflect the views of the funding agencies or the U.S. government.

Report Documentation Page

Form Approved
OMB No. 0704-0188

1. REPORT DATE	2. REPORT TYPE	3. DATES COVERED
27 JUL 2010		**00-00-2010 to 00-00-2010**

4. TITLE AND SUBTITLE	5a. CONTRACT NUMBER
Short Note on Complexity of Multi-Value Byzantine Agreement	5b. GRANT NUMBER
	5c. PROGRAM ELEMENT NUMBER
6. AUTHOR(S)	5d. PROJECT NUMBER
	5e. TASK NUMBER
	5f. WORK UNIT NUMBER
7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES) **University of Illinois at Urbana-Champaign,Department of Electrical and Computer Engineering,Coordinated Science Laboratory,Urbana,IL,61801**	8. PERFORMING ORGANIZATION REPORT NUMBER
9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)	10. SPONSOR/MONITOR'S ACRONYM(S)
	11. SPONSOR/MONITOR'S REPORT NUMBER(S)

12. DISTRIBUTION/AVAILABILITY STATEMENT
Approved for public release; distribution unlimited

13. SUPPLEMENTARY NOTES

14. ABSTRACT

15. SUBJECT TERMS

16. SECURITY CLASSIFICATION OF:			17. LIMITATION OF ABSTRACT	18. NUMBER OF PAGES	19a. NAME OF RESPONSIBLE PERSON
a. REPORT	b. ABSTRACT	c. THIS PAGE			
unclassified	**unclassified**	**unclassified**	**Same as Report (SAR)**	**5**	

1 Introduction

Inspired by [4], and the deterministic multi-valued Byzantine agreement algorithm in our recent technical report [5], we derive a randomized algorithm that achieves multi-valued Byzantine agreement with high probability, and achieves optimal complexity. The discussion in this note is not self-contained, and relies heavily on the material in [5] – please refer to [5] for the necessary background.

Consider a synchronous fully connected network with n nodes, namely $0, 1, \ldots, n-1$. Let node 0 be the source; the other $n-1$ nodes are called *peers*. At most $t < n/3$ nodes can be faulty. The goal here is for the $n-1$ peers to agree on the values sent by the source (similar to the Byzantine Generals problem in the work of Pease, Shostak and Lamport). This is also known as the "broadcast" problem. Our algorithm achieves agreement on a long message of l bits with high probability. Similar to the algorithm in [5], the proposed randomized Byzantine agreement algorithm progresses in generations. In each generation, D bits are being agreed upon, with the total number of generations being l/D. For convenience, we assume l to be an integral multiple of D.

2 Operations when no failure detected

Step 1: The source node 0 sends the D bits to each of the peers. The peers do not transmit.

Step 2: Every node i (including the source) computes $h_{i,j} = (K_{i,j}, H(m_i, K_{i,j}))$ for every $j \neq i$, where (i) $K_{i,j}$ is a randomly selected key of k bits, (ii) for $i \neq 0$, m_i represents the D bits of the current generation received by node i, and for $i = 0$, m_i represents the D bits of data that the source sends in step 1, and (iii) $H(m, K)$ is the almost-universal hash function of D/k bits introduced in [1]. For convenience, we assume D to be an integral multiple of k. Then node i sends $h_{i,j}$ to node j.

Step 3: Every fault-free node i, on receipt of $h_{j,i} = (K_{j,i}, h^*)$ from node j, computes hash of m_i using key $K_{j,i}$ and compares it with h^* in $h_{j,i}$. If any of these comparisons results in a mismatch, then node i has detected inconsistency (or misbehavior by a faulty node). After receiving the hash values from all the peers, each node i broadcasts a 1-bit notification, indicating whether all hash values are consistent with m_i or not, using a traditional Byzantine agreement algorithm – similar approach is used for the deterministic agreement algorithm in [5]. If no notification indicating inconsistency is received, then every node i decides on m_i, and the current generation completes. If any node indicates inconsistency detected, then the extended step (described below) is added.

According to [1], the probability for any distinct D-bit messages m_i and m_j to produce the same hash value $H(m_i, K) = H(m_j, K)$ for a random key K of k bits is upper bounded by $2^{-k}D/k$. For the fault-free nodes to decide on different values, at least one comparison of hash values must erroneously result in a match. Thus, the probability that the fault-free nodes will decide on different values is upper bounded by $2^{-k}D/k$. (A better bound on this probability can potentially be derived, but this bound suffices our purpose here.) Let us assume that $2^{-k}D/k < 1$, by proper choice of k.

Extended Step: In the extended step, every node broadcasts all the packets it has received or has sent in steps 1 and 2, using a traditional Byzantine agreement algorithm. Using these broadcast information, identical "diagnosis graphs" are formed at all fault-free nodes. The formation and use of the diagnosis graph here is the same as the algorithm in [5]. The reader is referred to [5] for the details of the diagnosis graph.

3 Operations after failure detected

After a failure is detected, and the extended step is finished, a new generation of D bits of new data begins. Let us say that nodes i and j accuse(trust) each other if edge ij is marked $f(g)$ in the diagnosis graph (see [5]). Since a fault-free node never accuses another fault-free node, a fault-free node can be accused by at most t other nodes. If a node is accused by more than t other nodes, this node is identified as faulty. If the source node 0 is identified as faulty, then the fault-free peers can terminate the algorithm and all agree on some default value. If a peer is identified as faulty, it is *isolated* (or removed) from the network, and the algorithm below is executed only by the remaining nodes. Now consider the case when the source node is accused by no more than t peers.

Step 1: Find a spanning tree routed at the source such that (i) the tree covers all nodes that have not been isolated, and (ii) it consists only g edges in the diagnosis graph. Then the D bits of data of the current generation is routed through this spanning tree.

Since the source node is not identified as faulty, it is connected to at least $n - t - 1 > 1$ peers, each with a g-edge directly. Then, to show that the required spanning tree always exists, it only left to show that if the source and all f-edges are removed, the remaining nodes that have not been isolated are all connected to each other. Consider any pair of peers i and j that are not identified as faulty. Consider two cases:

- Nodes i and j trust each other: Then edge ij between nodes i and j in the diagnosis graph must be a g-edge.
- Nodes i and j accuse each other: Since any node that is not yet isolated can accuse at most t nodes, nodes i and j each may accuse at most $t-1$ of the other $n-3$ peers. Thus, among the other $n-3$ peers, node i trusts at least $(n-3)-(t-1) = n-t-2 \geq (3t+1)-t-2 \geq 2t-1 \geq t$ peers (we assume $t \geq 1$). Since node j may accuse at most $t-1$ of the t other peers that node i trusts, it follows that there exists at least one other peer that i and j both trust. Thus, nodes i and j are connected in the diagnosis graph with a 2-hop path consisting of g-edges.

The rest of the algorithm is the same as the case when no failure is yet detected. As a clarification, note that in step 3, only nodes that are not isolated already may send or receive messages.

4 Security and Complexity Analysis

4.1 Security of the Algorithm

The security of the algorithm relies on the fact that the keys are not sent in step 3 until step 2 is complete. As seen in Section 2, the probability of the misbehavior by the faulty nodes being undetected is upper bounded by $2^{-k}D/k$. In other words, the misbehavior in a particular generation will be detected with probability at least $\rho = 1 - 2^{-k}D/k$. Then the probability that the misbehavior is always detected given that the faulty nodes misbehave in x generations is lower bounded by ρ^x, which is a decreasing function in x. Notice that if first $t(t+1)$ instances of misbehavior is detected, then all faulty nodes will be identified and isolated (by an "instance" we mean a generation in which at least one faulty node misbehaves by sending inconsistent data). Thus, the probability that the misbehavior is always detected, i.e., the probability of achieving agreement correctly on all l bits, is

$$\begin{aligned} P_{correct} \geq \rho^{t(t+1)} &= (1 - 2^{-k}D/k)^{t(t+1)} && (1)\\ &= 1 - O(t(t+1)2^{-k}D/k)) && (2)\\ &= 1 - O\left(n^2\, 2^{-k}D/k\right) && (3) \end{aligned}$$

4.2 Complexity of the Algorithm

Data transmissions: Every peer receives D bits through steps 1 and 2. So $(n-1)D$ bits are transmitted in each generation, which leads to $(n-1)l$ bits for data transmissions throughout the whole algorithm.

Hash keys: Every node that is not identified as faulty sends $k + D/k$ bits (k bit key, and D/k bit hash value) to every other node. So at most $n(n-1)(k+D/k)$ bits are transmitted in each generation, which leads to at most $n(n-1)(k+D/k)l/D$ bits throughout the whole algorithm.

Broadcasts in step 4: In step 4, every node broadcasts a 1 bit of notification. Let us denote B as the communication complexity of broadcasting 1 bit. Then the total cost for the broadcast in step 4 is nB bits, which lead to nBl/D bits over the whole algorithm.

Broadcasts in extended step: In the extended step, every node broadcasts D bits. Thus nDB bits are transmitted in each extended step. Since there will be at most $t(t+1)$ extended steps, the total cost of broadcasts in extended steps is at most $nDBt(t+1)$ bits throughout the whole algorithm.

Now we have a upper bound on $C(l)$, the total number of bits being transmitted to achieve agreement on l bits, as:

$$(n-1)l + n(n-1)(k+D/k)l/D + nBl/D + nDBt(t+1) \tag{4}$$

$$= (n-1)l + O(n^2kl/D + n^2l/k + nBl/D + n^3BD). \tag{5}$$

Notice that broadcast algorithm of complexity $\Theta(n^2)$ are known [2, 3], so we assume $B = \Theta(n^2)$. Then we have

$$C(l) = (n-1)l + O(n^2l/k + (n^2k + n^3)l/D + n^5D). \tag{6}$$

Then the per-bit communication complexity is

$$\alpha = C(l)/l = n - 1 + O(n^2/k + (n^2k + n^3)/D + n^5D/l). \tag{7}$$

4.3 Achieving high probability of agreement with low per-bit complexity

Now let us consider Equations 3 and 7 together. If we choose k and D such that the following conditions are all satisfied

- k and D are both unbounded increasing functions of l;
- $D = o(2^k k)$ and $D = o(l)$;
- $k = o(D)$,

then $P_{correct} \to 1$ and $\alpha \to n-1$ as l gets large. For example, we can choose $k = \log l$ and $D = l^{1-\beta}$ for some positive constant $0 < \beta < 1$, then

$$P_{correct} = 1 - O(n^2 l^{-\beta}/\log l) \tag{8}$$

$$\alpha = n - 1 + O(n^2/\log l + (n^2 \log l + n^3)l^{-(1-\beta)} + n^5 l^{-\beta}). \tag{9}$$

From [4, 5], we know that the per-bit complexity α is lower bounded by $n-1$. Thus, as l approaches ∞, the proposed algorithm achieves agreement of l bits with probability approaching 1, with per-bit complexity approaching the lower bound of $(n-1)$.

References

[1] Z. Beerliova-Trubiniova, M. Hirt, and M. Riser. *Efficient Byzantine Agreement with Faulty Minority.* Springer-Verlag, 2007.

[2] P. Berman, J. A. Garay, and K. J. Perry. Bit optimal distributed consensus. *Computer science: research and applications*, 1992.

[3] B. A. Coan and J. L. Welch. Modular construction of a byzantine agreement protocol with optimal message bit complexity. *Inf. Comput.*, 97(1):61–85, 1992.

[4] M. Fitzi and M. Hirt. Optimally efficient multi-valued byzantine agreement. In *PODC '06*, 2006.

[5] G. Liang and N. Vaidya. Complexity of multi-valued byzantine agreement. *Technical Report, CSL, UIUC*, June 2010.